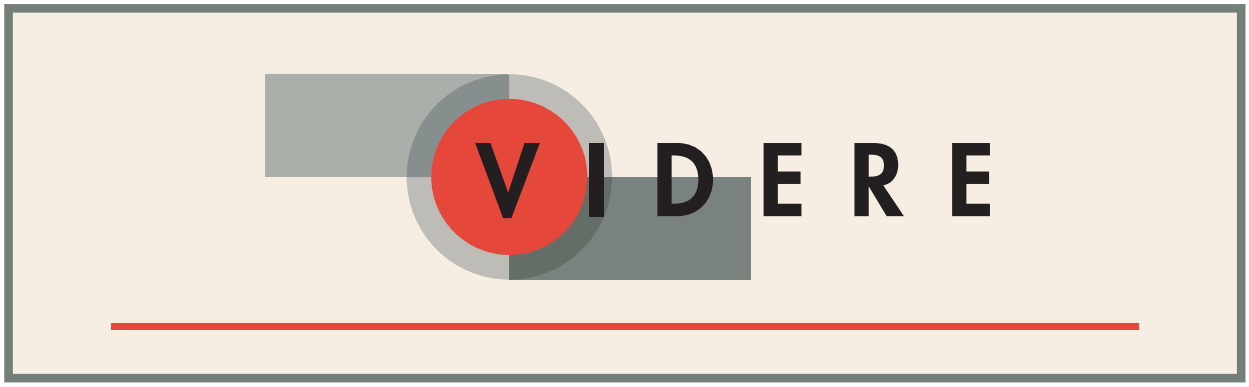
Toward a generic framework for recognition based on uncertain geometric features

Xavier Pennec



Videre: Journal of Computer Vision Research

Quarterly Journal

Winter 1998, Volume 1, Number 2

The MIT Press

Article 3

Toward a Generic Framework for Recognition Based on Uncertain Geometric Features

Xavier Pennec

Videre: Journal of Computer Vision Research (ISSN 1089-2788) is a quarterly journal published electronically on the Internet by The MIT Press, Cambridge, Massachusetts, 02142. Subscriptions and address changes should be addressed to MIT Press Journals, Five Cambridge Center, Cambridge, MA 02142; phone: (617) 253-2889; fax: (617) 577-1545; e-mail: journals-orders@mit.edu. Subscription rates are: Individuals $30.00, Institutions $125.00. Canadians add additional 7% GST. Prices subject to change without notice.

Toward a Generic Framework for Recognition Based on Uncertain Geometric Features

Xavier Pennec[1]

The recognition problem is probably one of the most studied in computer vision. However, most techniques were developed on point features and were not explicitly designed to cope with uncertainty in measurements.

The aim of this paper is to express recognition algorithms in terms of uncertain geometric features (such as points, lines, oriented points, or frames). In the first part we review the principal matching algorithms and adapt them to work with generic geometric features. Then we analyze some noise models on geometric features for recognition, and we consider how to cope with this uncertainty in the matching algorithms. We then identify four key problems for the implementation of these algorithms. Last but not least, we present a new statistical analysis of the probability of false positives that demonstrates a drastic improvement in confidence and complexity that we can obtain by using geometric features more complex than points.

Keywords: 3-D object recognition, invariants of 3-D objects, feature-matching, uncertain geometric features, generic features, matching error analysis

1. INRIA Sophia—Projet Epidaure, 2004 Route des Lucioles BP 93, 06902 Sophia Antipolis Cedex, France. Phone: +33 4 92 38 76 64, Fax: +33 4 92 38 76 69. xpennec@sophia.inria.fr, Web: www.inria.fr/epidaure/personnel/pennec/pennec.html

1 Introduction

The recognition problem is probably one of the most studied in computer vision. (See for instance [2, 8].) Many algorithms have been developed to compare two images or to recognize objects against an a priori model. These models are mostly constructed from point features and can vary according to rigid or affine transformations. One can cite for instance the Geometric Hashing [26], the ICP [3, 42], or the alignment algorithm [1, 20].

However, while these algorithms are efficient for well-scattered data with little noise, it becomes more and more difficult to find small objects in a complex and noisy scene. The maximal complexity of these algorithms is then reached and the problem of uncertainty handling becomes critical. Moreover, the geometric models of the real world often lead one to consider features that are more complex than points, such as lines [14], planes [10], oriented points, or frames [32, 34]. As we will show below, using these kinds of features directly in the recognition algorithm can lead to important improvements in complexity, accuracy, and robustness. On the other hand, we must be very rigorous in handling uncertainty to avoid false negatives and paradoxes [33].

The aim of this paper is express the above recognition algorithms in terms of uncertain geometric features. We restrict our analysis to the matching of similar features in the same space (3D-3D or 2D-2D for instance). In the following section, we briefly describe the theory already developed for handling the uncertainty of geometric features. In Section 3, we review the principal matching algorithms and adapt them to work with generic geometric features. In Section 4, we analyze three noise models for recognition (bounded, probabilistic, and a combination of both) and how to modify the matching algorithms to cope with uncertainty in features measurement. This leads to the identification of four key problems for the implementation in Section 5. In the last section, we analyze the drawbacks of uncertainty on recognition algorithm: false positives. We present a new method to evaluate qualitatively their probability of occurrence and show that using geometric features more complex than points drastically improves the algorithms' performance and robustness.

2 Geometric Features

We have shown in [33] that geometric features generally do not belong to a vector space but rather to a manifold and that this induces paradoxes if we try to use the standard techniques for points with them. For instance, we can represent a 3-D rotation by its matrix R, the corresponding unit quaternions $\pm q$, or the rotation vector $r = \theta \cdot n$. Using

the barycenter to compute the mean, we obtain either $\underline{R} = \frac{1}{n}\sum_i R_i$, $\underline{q} = \frac{1}{n}\sum_i q_i$, or $\underline{r} = \frac{1}{n}\sum_i r_i$. The three results correspond to different rotations.[1]

From a mathematical point of view, all the operations we define on features should rely on intrinsic characteristics of the manifold and not on the vector properties of some particular chart. Moreover, there generally is a transformation group acting on the manifold that models the possible image viewpoints and/or the subject movement in the images. Any operation should be invariant or "covariant" with respect to the action of this group. For instance, the barycenter of points is invariant under the action of rigid or affine transformations: $A(\sum x_i) = \sum Ax_i$. In the case of geometric features, these requirements are much more difficult to meet, and we have to look at rather deep results in differential geometry in order to understand the intrinsic characteristics of the manifold we can use.

In this section, we summarize the essence of the theory developed in [35]. More specifically, we present some intrinsic characteristics of the manifolds of geometric features and we show that most useful operations on such features can be expressed in this framework using a very small number of operations (atomic operations). The interested reader can consult [39, chap. 9], [25], and [7] for more theoretical results.

2.1 Riemannian Manifolds

In the geometric framework, one specifies the structure of a manifold $\mathcal{M}$ by a *Riemannian metric*. This is a continuous collection of dot products on the tangent space at each point x of the manifold. Thus, if we consider a curve on the manifold, we can compute at each point its instantaneous speed vector and its norm, the instantaneous speed. To compute the length of the curve, we can proceed as usual by integrating this value along the curve. The distance between two points of a connected Riemannian manifold is the minimum length among the curves joining these points. The curves realizing this minimum for any two points of the manifold are called *geodesics*. The calculus of variations shows that geodesics are the solutions of a system of second-order differential equations depending on the Riemannian metric.

In this article, we assume that the manifold is *geodesically complete*, i.e., that the definition domain of all geodesics can be extended to $\mathbb{R}$. This means that the manifold has no boundary nor any singular points that we can reach in a finite time. As an important consequence, the Hopf-Rinow-De Rham theorem states that there always exists at least one minimizing geodesic between any two points of the manifold (i.e., whose length is the distance between the two points).

Exponential chart From the theory of second-order differential equations, we know that there exists one and only one geodesic starting at a given feature x with a given tangent vector. This allows us to develop the manifold in the tangent space along the geodesics (think of rolling a sphere along its tangent plane at a given point). The geodesics going through this point are transformed into straight lines, and the distance along these geodesics are conserved (at least in a neighborhood of x).

1. The first two are not even rotations unless they are renormalized: the sum of orthogonal matrices is generally not an orthogonal matrix and the sum of unit quaternions is not a unit quaternion. Moreover, choosing the sign of the unit quaternion used in the sum is not a trivial problem.

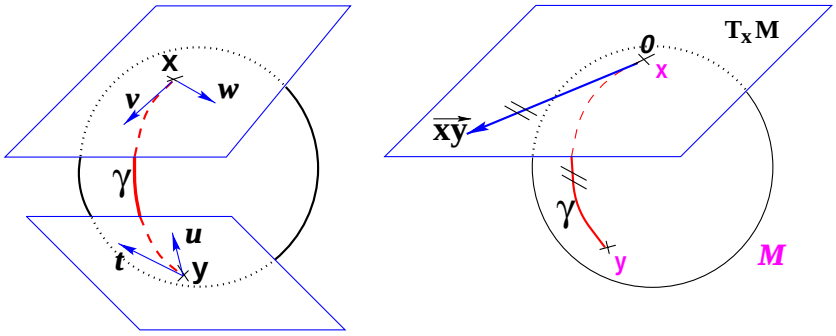


Figure 1. Left: The tangent planes at points x and y of the sphere $\mathcal{S}_2$ are different: the vectors v and w of $T_x\mathcal{M}$ cannot be compared to the vectors t and u of $T_y\mathcal{M}$. Thus, it is natural to define the dot product on each tangent plane. **Right:** The geodesics starting at x are straight lines in the exponential map, and the distance along them is conserved.

The function that maps to each vector the corresponding point on the manifold is called the *exponential map*.

This map is defined in the whole tangent space $T_x\mathcal{M}$ (since the manifold is geodesically complete), but it is one-to-one only locally around the origin. If we look for the maximal domain where this map is one-to-one, we find out that it is a star-shaped domain delimited by a continuous curve C_x called the *tangential cut-locus*. The image of C_x by the exponential map is the *cut locus* $\mathcal{C}_x$ of point x. Roughly, this is the set of points where several minimizing geodesics starting from x meet.[2] On the sphere $\mathcal{S}_2$ for instance, the cut locus of a point x is its antipodal point, and the tangential cut locus is the circle of radius π.

The exponential map within this domain realizes a chart called the *exponential chart*. It covers the whole manifold except for the cut locus of the development point, which has a null measure. Let $\vec{xy}$ be the representation of y in this chart. Then its distance to the origin is $\text{dist}(x, y) = \|\vec{y}\|_x$. This chart is somehow the "most linear" chart of the manifold with respect to the feature x: geodesics starting from this point are straight lines, and the distance is (locally) conserved along them.

2.2 Homogeneous Manifolds

Invariant metric Now, since we are working with features onto which acts a transformation group that models the possible image viewpoints, it is natural to choose an invariant Riemannian metric. (The existence conditions are detailed in [35].) This way, all the measurements based on distance are independent of the image reference frame or the transformation of the image: $\text{dist}(x, y) = \text{dist}(g \star x, g \star y)$.

Let o be a point of the manifold that we call the origin: we call *principal chart* the exponential chart at the origin for the invariant metric and we denote by $\vec{x}$ the representation of x in this chart. Assuming that we have chosen an orthonormal coordinate system, the distance with the origin is $\text{dist}(o, x) = \|\vec{x}\|$. Now, let $\mathrm{f_y}$ be a "placement function," i.e., a transformation such that $\mathrm{f_y} \star o = \mathrm{y}$. The distance between any two points is $\text{dist}(x, y) = \text{dist}\left(\mathrm{f_y^{(-1)}} \star \mathrm{x}, o\right) = \|\mathrm{f_{\vec{y}}^{(-1)}} \star \vec{\mathrm{x}}\|$. One can verify that this formula is independent of the choice of the placement function.

Exponential chart and map at other points The previous formula shows that the vector $(\mathrm{f_{\vec{y}}^{(-1)}} \star \vec{\mathrm{x}})$ is the representation of x in an exponential chart at feature y. The coordinate system of this chart is orthonormal, but it depends on the chosen placement function. It is often more interesting to use the nonorthogonal coordinate system induced by

2. More precisely, the cut locus is the closure of this set.

the principal chart, which is independent of the placement function: this is the vector $\overrightarrow{\mathrm{yx}} = J(\mathrm{f}_{\vec{\mathrm{y}}}) \cdot (\mathrm{f}_{\vec{\mathrm{y}}}^{(-1)} \star x)$, where $J(\mathrm{f}_{\vec{\mathrm{x}}}) = \left.\frac{\partial(\vec{\mathrm{f}} \star \vec{\mathrm{x}})}{\partial \vec{\mathrm{x}}}\right|_{\vec{\mathrm{x}}=o}$ is the Jacobian of the origin's translation in the principal chart.

From a practical point of view, this means that we can reinterpret the local calculations on points as local calculations in the principal chart of our manifold by replacing $x - y$ with $\overrightarrow{\mathrm{yx}}$ and $\mathrm{x} + \overrightarrow{\delta \mathrm{x}}$ with $\exp_{\vec{\mathrm{x}}}(\overrightarrow{\delta \mathrm{x}}) = \mathrm{f}_{\vec{\mathrm{x}}} \star (J(\mathrm{f}_{\vec{\mathrm{x}}})^{(-1)} \cdot S_{\vec{\delta \mathrm{x}}})$. For instance, the empirical covariance matrix with respect to a point y, becomes [35]:

$$\begin{aligned} \Sigma_{\vec{\mathbf{x}}\vec{\mathbf{x}}}(\vec{\mathrm{y}}) &= \frac{1}{n} \sum_{i=1}^{n} \overrightarrow{\mathrm{yx}_i} \cdot \overrightarrow{\mathrm{yx}_i}^{\mathrm{T}} \\ &= \frac{1}{n} J(\mathrm{f}_{\vec{\mathrm{y}}}) \cdot \left(\sum_{i=1}^{n} \left(\mathrm{f}_{\vec{\mathrm{y}}}^{(-1)} \star \vec{\mathrm{x}}_i \right) \cdot \left(\mathrm{f}_{\vec{\mathrm{y}}}^{(-1)} \star \vec{\mathrm{x}}_i \right)^{\mathrm{T}} \right) \cdot J(\mathrm{f}_{\vec{\mathrm{y}}})^{\mathrm{T}}. \end{aligned} \tag{1}$$

Atomic operations In fact, we have shown that most of the interesting operations for us on deterministic and uncertain geometric features can be expressed in the principal chart of the manifold with only the few operations we have introduced. From a computational point of view, this means that the only operations we have to implement for each type of feature are the action of a transformation $(\mathrm{f} \star \vec{\mathrm{x}})$ with its Jacobians $\frac{\partial(\mathrm{f} \star \vec{\mathrm{x}})}{\partial \mathrm{f}}$ and $\frac{\partial(\mathrm{f} \star \vec{\mathrm{f}})}{\partial \vec{\mathrm{x}}}$, and the placement function $\mathrm{f}_{\vec{\mathrm{x}}}$ with its Jacobian $\frac{\partial(\mathrm{f}_{\vec{\mathrm{x}}})}{\partial \vec{\mathrm{x}}}$. Every higher-level operation can be expressed as a combination of these and thus can be implemented independently of the considered type of feature. To simplify further computations, we can add to these atomic operations the Jacobian of the translation of the origin $J(\mathrm{f}_{\vec{\mathrm{x}}})$.

2.3 Practical Implementation

Computer model of random features Let x be a random feature. To define the mean value, we have to replace the standard expectation by the Fréchet expectation [33]:

$$\mathbb{E}\left[\, \mathbf{x} \,\right] = \arg\min_{\mathrm{y} \in \mathcal{M}} \left(\mathbf{E}\left[\, \mathrm{dist}(\mathrm{y}, \mathbf{x})^2 \,\right] \right).$$

Assuming that this mean value is unique, let $\mathbb{E}\left[\, \vec{\mathbf{x}} \,\right] = \vec{\bar{\mathrm{x}}}$ be its representation in the principal chart. Its covariance matrix in the same chart is given by

$$\Sigma_{\mathbf{xx}} = \mathbf{E}\left[\, \overrightarrow{\bar{\mathrm{x}}\mathbf{x}} \cdot \overrightarrow{\bar{\mathrm{x}}\mathbf{x}}^{\mathrm{T}} \,\right] = J(\vec{\mathrm{f}}_{\vec{\bar{\mathrm{x}}}}) \cdot \mathbf{E}\left[\, (\vec{\mathrm{f}}_{\vec{\bar{\mathrm{x}}}}^{(-1)} \star \vec{\mathbf{x}}) \cdot (\vec{\mathrm{f}}_{\vec{\bar{\mathrm{x}}}}^{(-1)} \star \vec{\mathbf{x}})^{\mathrm{T}} \,\right] \cdot J(\vec{\mathrm{f}}_{\vec{\bar{\mathrm{x}}}})^{\mathrm{T}}.$$

This information is often sufficient to model the random feature and provide a compact representation. Thus, from a computational point of view, we define a random feature by its approximation: $\mathbf{x} \sim (\vec{\bar{\mathrm{x}}}, \Sigma_{\mathbf{xx}})$. In this framework, a deterministic feature has a null covariance matrix. Random transformations are modeled similarly.

Basic operations Using this representation, the **action** of a random transformation $\mathbf{f} \sim (\vec{\bar{\mathrm{f}}}, \Sigma_{\mathbf{ff}})$ on a (random) feature $\mathbf{x} \sim (\vec{\bar{\mathrm{x}}}, \Sigma_{\mathbf{xx}})$ gives the random feature

$$\mathbf{y} = \mathbf{f} \star \mathbf{x} \sim \left(\vec{\bar{\mathrm{f}}} \circ \vec{\bar{\mathrm{x}}},\ \Sigma_{\mathbf{yy}} \right),$$

where $\Sigma_{\mathbf{yy}} = J_{\vec{\mathrm{f}}} \cdot \Sigma_{\mathbf{ff}} \cdot J_{\vec{\mathrm{f}}}^{\mathrm{T}} + J_{\vec{\mathrm{x}}} \cdot \Sigma_{\mathbf{xx}} \cdot J_{\vec{\mathrm{x}}}^{\mathrm{T}}$ with $J_{\vec{\mathrm{f}}} = \left.\frac{\partial(\vec{\mathrm{f}} \star \vec{\mathrm{x}})}{\partial \vec{\mathrm{f}}}\right|_{\vec{\mathrm{f}}=\vec{\bar{\mathrm{f}}}}$ and $J_{\vec{\mathrm{x}}} = \left.\frac{\partial(\vec{\mathrm{f}} \star \vec{\mathrm{x}})}{\partial \vec{\mathrm{x}}}\right|_{\vec{\mathrm{x}}=\vec{\bar{\mathrm{x}}}}$.

The **distance** between two features is simply

$$\mathrm{dist}(x, y) = \|\overrightarrow{\mathrm{xy}}\|_{\mathrm{x}} = \sqrt{\left(\vec{\mathrm{f}}_{\vec{\mathrm{x}}}^{(-1)} \star \vec{\mathrm{y}}\right)^{\mathrm{T}} \cdot \left(\vec{\mathrm{f}}_{\vec{\mathrm{x}}}^{(-1)} \star \vec{\mathrm{y}}\right)}.$$

Similarly, the Mahalanobis distance between a random feature $\mathbf{x} \sim (\bar{\mathrm{x}},\ \Sigma_{\mathbf{xx}})$ and a deterministic feature y becomes

$$\begin{aligned}\mu^2(\mathbf{x}, \mathrm{y}) &= \overrightarrow{\bar{\mathrm{x}}\mathrm{y}}^{\mathrm{T}} \cdot \Sigma_{\mathbf{xx}}^{(-1)} \cdot \overrightarrow{\bar{\mathrm{x}}\mathrm{y}} \\ &= (f_{\bar{\mathrm{x}}}^{(-1)} \star \vec{\mathrm{y}})^{\mathrm{T}} \cdot J(f_{\bar{\mathrm{x}}})^{\mathrm{T}} \cdot \Sigma_{\mathbf{xx}}^{(-1)} \cdot J(f_{\bar{\mathrm{x}}}) \cdot (f_{\bar{\mathrm{x}}}^{(-1)} \star \vec{\mathrm{y}}).\end{aligned}$$

The Mahalanobis distance between two random features is defined by

$$\mu^2(\mathbf{x}, \mathbf{y}) = \min_{\mathrm{z}} \left(\mu^2(\mathbf{x}, \mathrm{z}) + \mu^2(\mathbf{y}, \mathrm{z})\right).$$

Higher-level operations Based on the atomic and basic operations, one can define many higher-level operations on geometric features. For instance, we developed in [31] three gradient descent algorithms to compute the mean feature by minimizing the standard and Riemannian least-squares distance, the weighted Riemannian least-squares distance, and the Mahalanobis distance. We also develop in [30] similar algorithms for computing the optimal registration.

2.4 Summary of Geometric Features

In short, the important point to note is that geometric features usually belong to manifolds that are not vector spaces. However, developing the manifold along the geodesics onto its tangent space at the origin give a chart (the principal chart) which is almost linear: geodesics going through the origin are straight lines, and the distances are conserved along them. Using an invariant metric for the transformation group acting on features, we can translate this chart at any point of the manifold. From a practical point of view, this means that we can reinterpret the local calculations on points as local calculations in the principal chart of our manifold by replacing $\overrightarrow{yx} = x - y$ with $\overrightarrow{\mathrm{yx}} = J(\mathrm{f}_{\vec{\mathrm{y}}}) \cdot (\mathrm{f}_{\vec{\mathrm{y}}}^{(-1)} \star \mathrm{x})$, and $x + \overrightarrow{\delta \mathrm{x}}$ with $\exp_{\vec{\mathrm{x}}}(\overrightarrow{\delta \mathrm{x}}) = \mathrm{f}_{\vec{\mathrm{x}}} \star (J(\mathrm{f}_{\vec{\mathrm{x}})}^{(-1)}) \cdot \delta \mathrm{x}$.

It turns out that all the interesting operations on deterministic and probabilistic features and transformations can be expressed in the principal chart using only a few atomic operations and their Jacobians: the composition and inversion of transformations ($\mathrm{f} \circ \mathrm{g}$ and $\mathrm{f}^{(-1)}$), the action of a transformation ($\mathrm{f} \star \vec{\mathrm{x}}$), and the placement function ($\mathrm{f}_{\vec{\mathrm{x}}}$). On this basis, we can construct many higher-level algorithms, such as the computation of the mean feature or the computation of the transformation between two sets of matched features (registration).

Example of features In [35], we have implemented this framework for 3-D rigid transformations acting on different kinds of features such as frames, semi-oriented frames, and points. Frames are composed of a point and an orthonormal trihedron and are equivalent to rigid transformations. The principal chart is composed of the rotation vector representing the trihedron (or the rotation) and a vector representing the point position (or the translation). Semi-oriented frames model the differential properties of a point on a surface. They are composed of a point

and a trihedron (t_1, t_2, n), where $(t_1, t_2) \equiv (-t_1, -t_2)$ are the principal directions of the surface.

3 Matching Algorithms

In this paper, we consider that *matching* is aimed at finding correspondences between two set of features, whereas *registration* is the computation of the transformation between two matched sets of features. These two problems are often intrinsically linked to constitute the *feature-based recognition* problem. Here, we focus on matching methods.

We have one set $\mathcal{X} = \{\mathrm{x}_1, \ldots \mathrm{x}_m\} \in \mathcal{M}^m$ of m features modeling the first image and similarly a set $\mathcal{Y} = \{\mathrm{y}_1, \ldots \mathrm{y}_n\} \in \mathcal{M}^n$ of n features modeling the second image. To distinguish the two sets, we use the vocabulary from the model-based recognition: we call *model* the set $\mathcal{X}$, as if it was the model of an object stored in a library, and *scene* the set $\mathcal{Y}$, in which we are looking for objects.

If we had two ideal acquisitions of the same object (with also perfect low-level processing), the model and the scene would have the same number of features and the two sets would be simply transformed into one another using a transformation $\mathrm{f} \in \mathcal{G}$, with possibly a permutation in index of features. The matching problem is in this case very simple: find the index permutation that identifies features up to a transformation f. In practice, this is not so easy: the object we are looking for is usually not the only one in our images, and hence there are numerous extra features in the two images that do not have to be matched but that we cannot suppress a priori. This is called *clutter*. Moreover, even if an object is perfectly known (from its CAD model for instance), an image can be noisy enough to inhibit the detection of some features. We call this *occlusion* as in computer vision, even if the phenomenon is different in other domains such as 3-D imaging. Last but not least, the measure of features is inherently noisy and the superimposition of matched features after registration will never be perfect.

In general, one considers that an object is *recognized* if there exists a transformation $\mathrm{f} \in \mathcal{G}$ that superimposes (within a given error) a sufficient number of features in the two images. Thus, we want to maximize both the number of matches and the quality of these matches, that is the goodness of fit after registration.

Let π be the *matching function* that associates to the index i of a model feature x_i the index $j = \pi(i)$ of the matched feature y_j in the scene or the null feature $*$ if it is not matched. This function can thus be considered as an application from $\{1 \ldots m\}$ to $\{1 \ldots n+1\}$ or from $\mathcal{X}$ to $\mathcal{Y} \cup \{*\}$. There are $(n+1)^m$ such functions in the correspondence space Π. Some constraints can be added, such as the symmetry of matches. The basic idea in matching is to maximize the number of matches, but matches should also be consistent with the configuration of features. One often uses the following simple *plausibility function* to estimate the likelihood of a match: $\alpha(\mathrm{z}, \mathrm{z}') = 1$ if $\mathrm{dist}(\mathrm{z}, \mathrm{z}') \leq \varepsilon$ and 0 otherwise. The matching function is thus defined as the one that maximizes the matching score ($\mathrm{f} \star \mathrm{x}$ is the action of the transformation f on the feature x):

$$\hat{\pi} = \arg\max_{\pi \in \Pi} \left(\sum_i \alpha \left(\mathrm{f} \star \mathrm{x}_i,\ \mathrm{y}_{\pi(i)} \right) \right). \tag{2}$$

At the same time, we want to optimize the quality of matches and find the transformation that superimposes the model onto the scene. This is often done using least squares, with the convention that $\mathrm{dist}(\mathrm{x}, *) = 0$.

$$\hat{\mathrm{f}} = \arg\min_{\mathrm{f}\in\mathcal{G}} \left(\sum_i \mathrm{dist}\left(\mathrm{f}\star \mathrm{x}_i,\ \mathrm{y}_{\pi(i)}\right) \right). \tag{3}$$

There is thus a joint search in the correspondence space (find $\pi \in \Pi$) and in the transformation space (find $\mathrm{f} \in \mathcal{G}$). Each of these problems taken independently is relatively simple, but solving both together is much harder. In the remainder of this section, we investigate in the sequel the main algorithms used in image processing to solve this problem and how to formulate them in terms of features.

3.1 Interpretation Trees

If we look at the equations, we find that searching for the transformation f is linked to the matching solution π. The interpretation trees method proposed in [13] relies on the dual method: looking for π in the correspondence space Π and computing a posteriori the transformation f to validate the interpretation. The basic algorithm is thus to assume at each node of the interpretation tree the match of an unused feature in $\mathcal{X}$. When we arrive at a leaf we validate or discard this interpretation by looking for the transformation.

Several classical methods in artificial intelligence allow one to prune the search in this exponential tree ($(n+1)^m$ leaf). One of the most interesting is the introduction of geometric constraints with *unary invariants*. Assume that our features are segments: the length of a segment in the scene can only be inferior (modulo the error) to the length of the model segment since the only modifications on the length are due to occlusion and measurement errors: this is a unary constraint on the matches. Now if two segment matches satisfy the unary constraints, the angle between these segments in the model and in the scene should be almost identical: this is a *binary constraint of invariance*. For each type of feature, we can similarly develop a set of unary, binary, and possibly higher-order geometric constraints that will prune the interpretation tree and guarantee a local consistency during the depth-first search. The search of the transformation will actually give the proof of the global consistency on leaves. The introduction of measurement error is quite simple in this scheme: we just have to propagate the error on features measurement in the constraints computation. A complete study of these kind of techniques is developed in [13], but the complexity remains $O((n+1)^m)$ in the worst case.

3.2 Iterative Closest Point

This algorithm was introduced in [3] and [42]. It consists of the alternate optimization of the matches and the transformation. Since matches are computed from the distance between points, this algorithm can be easily generalized to features: from an initial transformation f_o, one iterates the two following steps:

- **Matching**: Each model feature x_i is matched to its nearest neighbor (NN) y_j in the scene $\mathcal{Y}$:

$$\pi_t(j) = \arg\min_j \mathrm{dist}\left(\mathrm{f}_t \star \mathrm{x}_i, \mathrm{y}_j\right), \qquad \text{i.e.,} \qquad \mathrm{y}_{\pi_t(j)} = \mathrm{NN}_{\mathcal{Y}}(\mathrm{f}_t \star \mathrm{x}_i).$$

- **Registration**: This is usually done using least squares. We have developed in [35] such methods on generic geometric features.

The above matching criterion is generally not symmetric: we can have $y_j = NN_{\mathcal{Y}}(x_i)$ while the reverse is false. To solve this problem, we have introduced in [32] a symmetric version of the nearest neighbor where we match x_i to $y_j = NN_{\mathcal{Y}}(x_i)$ if and only if $NN_{\mathcal{X}}\big(NN_{\mathcal{Y}}(x_i)\big) = x_i$.

The usual problem of this algorithm is to choose a termination criterion. Since we are using identified features (and not "continuous" features such as points on curves or surfaces), the set of possible matches is discrete, and each interpretation corresponds to a unique transformation. When the algorithm converges, we thus obtain not only the same matches but also the same transformation in successive iterations. This last characterization is computationally cheap and constitutes the termination criterion. (Adding a maximal number of iterations is also useful in case the algorithm does not converge.)

This algorithm is very easy to implement, but it needs a very fast nearest neighbor search in a manifold and an efficient registration computation. Moreover, it can be very sensitive to the initial transformation, and it tries to match the whole model with the whole scene. Thus, unless it is initialized with a very good transformation, it cannot find a very small set of good matches among a lot of clutter (like in proteins). For these reasons, it is often used as a *verification* algorithm after one of the other algorithms presented here.

3.3 Hough Transform

This method was introduced by Paul Hough in 1962 to detect simple geometric objects like lines from points by accumulating evidence in their parameter space. The method was generalized for recognition in the following way: let k be the minimum number of matches needed to compute a unique transformation (or at least a finite number) between the model and the scene. For each k-tuple of matches, we compute this transformation (if it exists), and we accumulate evidence in the transformation space to find the one that maximizes the number of matches (and hence the number of k-tuples matches).

The accumulation stage can be done with a discretization of the transformation space, which serves as an accumulator for the votes cast by the above k-tuple transformations. A sweeping stage is then necessary at the end to find the best transformation hypotheses. One can also keep all the computed transformations in a list and use a clustering algorithm (like in [11, 32]). An adaptive subdivision of the transformation space was proposed in [5]. At each step, the region is either subdivided or dismissed based on upper and lower bounds on the number of possible matches in the region. However, the linear equations used to efficiently compute these bounds seem difficult to generalize to generic geometric features.

The Hough transform was particularly studied for the inclusion of measurement errors; a synthesis can be found in [13]. The complexity is $O(m^k \cdot n^k)$, but we need to store the transformation space in memory and sweep it at the end of the algorithm to find the maximum. This can be a serious drawback when it comes to 3-D since rigid body motions have a dimension 6 and affine transformations a dimension 9.

3.4 Alignment or Prediction-Verification

This technique can be seen as a compromise between interpretation trees and Hough transform: one first hypothesizes a k-tuple of matches that allows one to compute a transformation. Then this hypothesis is verified

by mapping the model onto the scene and searching for new matches in a given error zone around the model features. The quality score of this hypothesis is simply the number of matches found, or it can be more elaborate. This technique was principally developed in [1, 20, 21].

In theory, one should repeat this scheme for all possible k-tuple of matches and keep the best hypotheses. For instance with 3-D points, we need three matches to specify a rigid transformation ($k = 3$). Thus, there are $C_m^3 = \binom{m}{3} = \frac{m!}{3!\cdot(m-3)!}$ ways to choose three points among m in the model, $C_n^3 = \binom{n}{3}$ ways in the scene, and 3! to match the two triplets. Therefore, we have $\binom{m}{3} \cdot \binom{n}{3} \cdot 3! = O(m^3 \cdot n^3)$ alignments or hypotheses to verify. In practice, one stops as soon as we estimate that an object is recognized and registered. Moreover, we can use the three invariants of the triplet (the distances between points) to index it in a hash table. This allows one to retrieve in quasi-constant time the compatible triplets in the scene. The complexity falls thus to $O(m^3 + n^3)$ for the prediction step. This use of the invariants of the k-tuple can be generalized to geometric features. (See Section 3.6.)

The verification step is particularly important since it should reject the bad hypotheses but keep the good ones. (See Section 6.) It is generally realized with an iterative nearest neighbor with a threshold on the distance for matching. Refining both the transformation and the matches is important if we want to stop as soon as we find a good solution. If we test all hypotheses, we can refine only the best ones.

3.5 Geometric Hashing

This technique was introduced by [26, 41]. The idea is to pre-compile the information about model objects in a hash table with an invariant representation (with respect to the action of the transformation group $\mathcal{G}$). This representation should also be redundant and based on local features to allow for the recognition in the presence of occlusions. At recognition time, we need to compute only the corresponding representation of the scene and accumulate evidence for the matching part of the scene with an object.

More precisely, the geometric hashing concerns the case of a subgroup of affine transformations acting on k-D points: one can then define an intrinsic coordinate system (or basis) of a model by choosing at most $k + 1$ of its points and expressing the coordinates of the others in this basis. For instance, two points are sufficient to define an orthonormal basis in 2-D (in fact a point and a direction are enough), or three non-aligned points in 3-D. We need exactly three points for an affine basis in 2-D and four points in 3-D. The coordinates of other points in this basis are invariant with respect to a global motion of the object.

The idea is to index in a pre-processing step all possible basis in the model by the coordinates of the other model points. (See Figures 4 and 5 for examples of hash tables.) At recognition time, one chooses an image basis and computes the coordinates of the other points in it. These coordinates being invariant, we retrieve for each point (thanks to the hash table) the model bases having a point in the same configuration, and we tally a vote for the matching of these model bases with the current scene basis. If the scene basis belongs to an indexed object, the number of votes for the corresponding model basis will be the number of points of the object that are visible in the scene (minus the points of the basis). The complexity is thus $O(n^{k+1})$ in the worst case for the

recognition if the access time of a hash table bucket is constant. This is obtained by considering n votes for every of the n^k possible scene bases. The verification step is not taken into account here.

The use of a hash table gives a sublinear recognition complexity with respect to a library of objects: one simply indexes all the objects in the same hash table and adds in the information stored to which object each basis belongs.

3.6 Geometric Invariant Indexing

The generalization of geometric hashing to geometric features other than points is not so easy, contrary to the alignment or the Hough transform methods. Indeed, the notions of basis and coordinates of other points in such a basis are linked to the structure of vector space, which is generally not available for geometric features. However, one can conceive a similar algorithm using the invariants of a fixed number of features instead of coordinates, the accumulation still taking place in the correspondence space.

If we take for instance 3-D points, the equivalent of the geometric hashing algorithm would be to use in the hash table the invariants of an ordered 4-tuple of points (x_i, x_j, x_k, x_l) instead of the coordinates of x_l in a basis constructed from (x_i, x_j, x_k). For the accumulation, it then seems natural to replace the increment of a basis match by the increment of all four individual matches: if (y'_i, y'_j, y'_k, y'_l) is a compatible 4-tuple of points in the scene, we will now vote for each match (x_i, y'_i).

After this modification, it is no longer necessary to impose the use of a 4-tuple of points: we can use a triplet of points or simply a pair. The complexity is in this case reduced from $O(n^4)$ to $O(n^3)$ or $O(n^2)$. However, we must be careful that decreasing the number of points drastically reduces the selectivity of invariants, and the probability of false positives becomes high very fast with the noise. The results of such an algorithm can thus be absurd. (See Section 6.) If we consider a triplet of points, we obtain an algorithm that is very close to the Hough transform, except that the accumulation is in the correspondence space instead of the transformation space. It is possible to combine both approaches by clustering for instance the transformations associated with each possible match.

This type of algorithm has been used in [6] with complex features (points with local shape descriptors), in [24, 18] for 3-D curves using differential invariants (principal directions and curvatures), in [40] for 3-D range data, and in [11, 12] to reduce the complexity of 3-D point-based matching from $O(n^4)$ to $O(n^3)$. In this last case, a pseudo-basis constituted of two points was replacing the standard three points basis for accumulation. We have also used such an algorithm in [19, 32] with 3-D frame features[3]: we used in this case a pair of frames to index, the invariants of such a pair being the rigid transformation from one frame to another. With this 6-D invariant space, we have a quite selective scheme and a complexity of $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n^4)$ for points. However, for the general feature case, the main problem is how to compute the invariants of a k-tuple of features, which will be detailed in Section 5.1.

3. A frame is a point with an orthonormal trihedron and is equivalent to a Euclidean coordinate system.

4 Error Handling

All these recognition algorithms are quite simple in the theoretical case of exact measurements. In practice, all our measurements are subject to errors due for instance to the deformations of the acquisition system and the image sampling in pixels and gray levels. The algorithms have to be adapted to explicitly take into account the fact that the features are inherently uncertain. For instance, in geometric hashing the measurement error propagates into the computation of invariants, and voting punctually in the hash table (just through the bin corresponding to the computed invariant value) could lead us to miss a great number of matches. We would then obtain *false negatives* which means that the object is not recognized when it should be. The same thing can happen with the alignment algorithm if the area search for a correspondent is too small with respect to the measurement error of the features enlarged by the uncertainty of the hypothesis transformation.

In this section, we analyze how measurement uncertainty should be taken into account in order to avoid false negatives. This will ensure the correctness of the recognition algorithms: objects will be recognized if they are present in the scene.

4.1 Error or Compatibility Zones

To handle uncertainty, we assume that we know an estimation of the measurement errors either as a bound on the values or as a probability law.

Conservative error bounds The approach of Grimson and Huttenlocher [14, 17] or Lamdan and Wolfson [27] is to propagate an error bound in the invariant computation to obtain an error zone for the votes in geometric hashing or for the search zone during alignment. Using this error zone, we are ensured not to miss a possible match. It is not assumed in this method that the error distribution is uniform but only that the distribution has a compact support that is included in the error zone. To be sure that we do not miss a possible match, we want to compute a conservative error bound (an upper bound) during the operations we perform on features. This fact ensures the correctness of the algorithm. On the other hand, the recursive use of upper bounds ends by giving error zones that are much larger than what is observed.

To illustrate this point, we have computed in [29] the conservative propagation of error bounds for rigid alignment and geometric hashing of 3-D points (Figure 2). We can see that the predicted error bound is much larger than the values actually observed. In fact, we have established in this case that

$$\text{Mean error} = \frac{\text{Predicted error}}{13.2} \quad \text{and} \quad \text{Max error} < \frac{\text{Predicted error}}{2}.$$

The conservative bound for this relatively simple example is thus already two times larger than the bound statistically observed. Moreover, the computation of these conservative bounds is especially difficult, and each operation needs a particular hand derivation of the bound.

Probabilistic error Another approach is to consider that we observe the realization of some random vectors and propagate the first moments of the distributions through all computations. This is especially well adapted to introduce a quantitative estimation of the quality of the

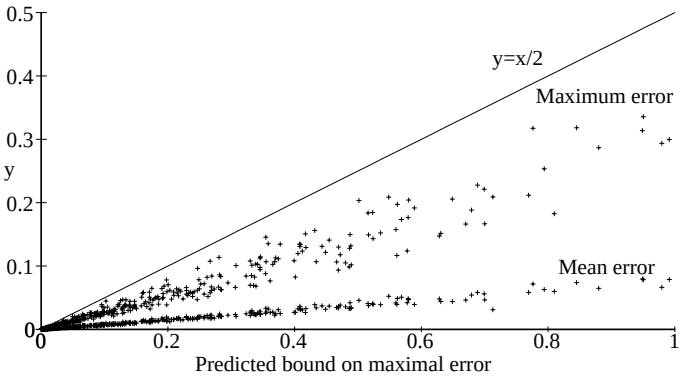


Figure 2. Maximal and mean observed error for 3-D points due to the uncertainty of the trihedron in the basis for geometric hashing. Each of the 500 values is computed as the mean and maximal error for 1000 perturbations of a three-point basis. The x value is the corresponding predicted maximal error.

matches in the matching criterion. This is the core of the probabilistic geometric hashing proposed by Rigoutsos and Hummel [37, 38].

According to the theory developed in the introduction, a random feature is approximated by its mean and its covariance matrix: $\mathbf{x} \sim (\overline{\mathrm{x}},\ \Sigma_{\mathbf{xx}})$. However, most probabilistic recognition methods assume that the error distribution is Gaussian to combine probabilities in the matching score. This assumption is consistent with our approximation since the Gaussian is the distribution that minimizes the information when we only know the mean and the covariance.[4]

However, the main problem of the Gaussian distribution is that its support is infinite. Thus, there is an always non-zero probability of matching any two features. This is catastrophic for the complexity of the matching algorithms since we should theoretically explore the whole correspondence space.

Truncated probabilistic error The usual solution [37] is to combine the two approaches by bounding the admissible error using the Mahalanobis distance. (This corresponds to a χ^2 in the Gaussian hypothesis.) The *error or compatibility zone* around feature $\mathbf{x} \sim (\overline{\mathrm{x}},\ \Sigma_{\mathbf{xx}})$ is thus:

$$\mathcal{Z}_\nu(\mathbf{x}) = \left\{ \mathrm{z} \in \mathcal{M} / \mu^2(\mathbf{x}, \mathrm{z}) = \vec{\bar{\mathrm{x}}\mathrm{z}}^{\mathrm{T}} \cdot \Sigma_{\mathbf{xx}}^{(-1)} \cdot \vec{\bar{\mathrm{x}}\mathrm{z}} \leq \nu^2 \right\}, \tag{4}$$

where ν^2 is a (generally global) threshold that can be interpreted in the Gaussian case as a χ^2. It is interesting to compare with the previous bounded error model:

$$\mathcal{Z}_\varepsilon(\mathbf{x}) = \left\{ \mathrm{z} \in \mathcal{M} / \ \mathrm{dist}(\overline{\mathrm{x}}, \mathrm{z})^2 = \vec{\bar{\mathrm{x}}\mathrm{z}}^{\mathrm{T}} \cdot \vec{\bar{\mathrm{x}}\mathrm{z}} \leq \varepsilon^2 \right\} \tag{5}$$

where the threshold ε is metric and thus harder to choose than a threshold without dimension. The truncated probabilistic model can thus be seen as a generalization of the bounded error model where the information matrix $\Sigma_{\mathbf{xx}}^{(-1)}$ is used as a local metric. Moreover, one can show [10, chap. 5, p. 152] that propagating covariance matrices using the Jacobians can be interpreted in a *deterministic* way as the *first-order approximation* of the error bound propagation.

4. By the way, it is using this very property that we have defined the Gaussian distribution on a manifold in [35].

4.2 Handling Uncertainty in Matching Algorithms

We describe in this section how to modify the recognition algorithms in order to make them handle an a priori known uncertainty on features. It is, however, desirable to verify a posteriori after recognition and registration that our guess about feature uncertainty is good. One can also think of iterating the whole process (matching, registration, and estimation of the noise on features).

Interpretation trees To handle uncertainty in this algorithm, we just have to propagate the uncertainty in the computation of unary, binary, and higher-order invariants and perform the constraint satisfaction test using a Mahalanobis distance between uncertain invariants.

Hough transform Assume that we have found a k-tuple of model features and a k-tuple of scene features that have compatible invariants. (This can be done using an invariant indexing technique.) We compute the transformation $\hat{\mathbf{f}}$ between these k-tuples and its uncertainty $\Sigma_{\mathbf{ff}}$. We obtain in fact a probabilistic transformation $\mathbf{f} \sim (\hat{\mathbf{f}},\ \Sigma_{\mathbf{ff}})$. To verify if these k matches are correct, we use the Mahalanobis distance between matched features after registration; but since the transformation is computed by minimizing these distances, we should not take into account the uncertainty of the transformation to obtain an unbiased result. (Otherwise the Mahalanobis distances would be underestimated.) This means that we test if $\mu^2(\hat{\mathbf{f}} \star \mathbf{x}_i, \mathbf{y}_i) \leq \nu^2$ for each of the k matches.

Since we need at least k matches to compute the transformation, this one is not reliable if at least one of the k tests fails. We reject in this case the transformation. If all tests are passed, the transformation is a possible one, and it is added in the accumulator *with its uncertainty*.

There are two main techniques for the accumulation. The most widely known is to vote for the bin where our transformation falls in the sampled transformation space. This is the origin of the term "accumulator." With the error handling, we now have to vote for all the bins of the accumulators that intersect the compatibility zone $\mathcal{Z}_\nu(\mathbf{f})$ of our probabilistic transformation. This problem is not so easy to solve efficiently and will be detailed in Section 5.4. We also have to sweep the sampled transformation space at the end of the algorithm to find the maximum score transformations. An alternative is to maintain during the algorithm a list of the best transformations. (The question is then how many do we have to keep.)

The second technique is to add each possible transformation $\mathbf{f}$ to a list and afterward use a clustering algorithm to extract and merge the sets of compatible transformations. We used a similar technique in [32].

Alignment The computation of the transformation between k features and the verification of theses k matches is exactly the same as for the Hough transform. The difference is that we now verify the hypothesized transformation by looking for additional matches with the other features. In this process, the uncertainty of the transformation has to be taken into account. One then searches for each model feature $\mathbf{x}_i$ the nearest neighbor $\mathbf{y}_j = \mathrm{NN}(\mathbf{f} \star \mathbf{x}_i)$ in the scene (according to the canonical or the Mahalanobis distance and possibly with the symmetry constraint on the nearest neighbor). The match is accepted if the following test is passed:

$$\mu^2(\mathbf{f} \star \mathbf{x}_i, \mathbf{y}_j) < \nu^2.$$

If the number of matches is sufficient for the hypothesized transformation, it is interesting to verify the global consistency of matches and an improvement of the transformation. We can indeed seriously reduce the transformation uncertainty using all matches.

ICP and global consistency verification If we have the matches (as with alignment), the following step is to recompute the transformation minimizing the Mahalanobis distance between all these matches. To improve robustness, we can rule out the outliers by verifying once again the Mahalanobis distance of the matches, but, since the new transformation is computed from these very matches, one has to do it *without the transformation uncertainty* to be unbiased. The test is thus for each match:

$$\mu^2(\hat{\mathbf{f}} \star \mathbf{x}_i, \mathbf{y}_j) < \nu^2.$$

The process can be iterated until convergence. To realize a real ICP from this verification scheme, it is sufficient to recompute the nearest neighbor of each model feature, keeping only those who pass the above test.

Hashing and geometric invariant indexing For geometric invariant indexing, we need to propagate the feature uncertainty in the invariant computation (as for the interpretation trees), and index these probabilistic invariants *with their error zone*. This last step raises the same problems as for the indexing of probabilistic transformations in the Hough transform. It will be detailed in Section 5.4.

Handling error in the geometric hashing algorithm is more complex. Consider for instance the case of 3-D points: we need first to fix the way we compute the basis from three points and estimate the uncertainty of this local basis (i.e., the uncertainty of the transformation $\mathbf{f}$ from the canonical basis to the local one) with respect to the uncertainty of the three points. The best way to do this is unclear, even if a least squares (between the three points and three fixed points in the canonical basis) seems to be adapted. We have in this case already developed the algorithm to get the registration and its uncertainty. Computing the coordinates of other points in the local basis then corresponds to the action of the probabilistic transformation $\mathbf{f}$. In this process, we must be careful to take into account the uncertainty of the transformation while computing the uncertainty of the invariant coordinates (otherwise a slightly large error of the basis features would rule out most of the matches). The last modification is to index (and retrieve) as above our probabilistic invariant coordinates *with their error zones*.

5 Some Key Problems

Up to now, we have formalized the main matching algorithms used on points in terms of features and seen how to modify them in order to explicitly handle uncertain measurements. In this section, we investigate some key problems that are raised by these algorithms and that constitute the difficult part to implement either due to the fact that we are using features or simply because of the uncertainty.

We have identified four main problems. For ICP, we know how to compute the transformation and its uncertainty, but the problem is to find efficiently the closest neighbor in a manifold (with a non-Euclidean Riemannian metric). A related problem for the Hough transform is to

cluster the transformations. Of course, the clustering of geometric features can be of use for many other important algorithms. The other algorithms (including the Hough transform) use more or less unary, binary, or higher-order *invariants*. The problem is not only to compute them but also to compare them efficiently and reliably. Last but not least, hashing is widely used to improve the algorithmic search for compatible features or invariants. How can we do this in presence of error and in a manifold?

5.1 n-ary Invariants: Shape Space

We have previously talked about unary, binary, or higher-order invariants as the characteristic invariants of the shape of k ordered features. If we can easily see that this notion corresponds to the distance for a pair of (rigid) points and, for instance, the three interpoint distances in a triplet of (still rigid) points, it is much harder to imagine what are the similarity invariants of a 4-tuple of points or the affine invariants of five oriented points.

We believe that an approach based on the shape theory is well adapted to tackle this problem. This theory was principally developed by Kendall and Le [23, 28] on points under similarities and rigid transformations. The idea is to characterize the configuration space of a set of k ordered features. By configuration, we mean what is invariant: the *shape*. The method is the following: a k-tuple of features is an element of $\mathcal{M}^k$. To obtain its shape, we identify all k-tuples that can be identified using a suitable transformation $\mathrm{f} \in \mathcal{G}$. The k-shape space is thus the quotient space $\mathcal{I}_k = \mathcal{M}^k/\mathcal{G}$ that Kendall denotes by $\Sigma(\mathcal{M}, \mathcal{G}, k)$. One can also see this as the "factorization" of a k-tuple of features into a shape $i \in \mathcal{I}_k$ and a "position" $\mathrm{f} \in \mathcal{F} \subset \mathcal{G}$ of this shape in space.

With this approach, the first problem is to find a metric on the shape space compatible with our metric on the manifold $\mathcal{M}$ and on the group $\mathcal{G}$. Roughly, this means that we want to have the same measurements on invariants independently of the position of the original k-tuple in space, and this is equivalent to finding a "factorization" of the metrics such that $\mathcal{M}^k = \mathcal{G} \times \mathcal{I}_k$. With the metric on the invariant space, we can determine geodesic and the exponential chart (the "most linear chart" with respect to the metric) at each point of the manifold.

Then, from a theoretical point of view, we can compute the mean invariant and its covariance matrix and the Mahalanobis distance. From a computational point of view, we need to find a way to implement the exponential chart at each point of $\mathcal{I}_k$ as we no longer have a *placement function* as in the homogeneous manifolds to identify things at feature x with things at the origin. The problem is however simpler than for features, since the basic operations on probabilistic invariants are reduced to

- Translation between a k-tuple of features and the pair k-shape/position: $\mathcal{X} \in \mathcal{M}^k \leftrightarrow (\mathrm{i}, \mathrm{f}) \in \mathcal{I}_k \times \mathcal{G}$.
- Distance between two k-shapes: $\mathrm{dist}(\mathrm{i}_1, \mathrm{i}_2)$.
- Mahalanobis distance between two probabilistic k-shapes: $\mu(\mathbf{i}_1, \mathbf{i}_2)$.

From these basic operations, we can construct algorithms to merge invariants, estimate the mean, etc. For efficient matching algorithms, we also need to find the nearest neighbor in this manifold or index these uncertain invariants.

5.2 Clustering of Uncertain Geometric Features

We are interested here in the clustering of transformations for the Hough transform. However, since the transformation group is a manifold, a more general problem is the clustering of geometric features on a manifold. The problem is somehow simplified with respect to the standard clustering problem since we have an uncertainty estimation on our features: we know thus the "scale" of the clusters. On the other hand, we generally do not know the number of classes.

It is possible to generalize some classical techniques based on the distance of points (see for instance [9, 22]) by replacing the distance between points with the Mahalanobis distance between uncertain features. On the contrary, space sampling techniques are less likely generalizable since we no longer have a vector space. (See also Section 5.4.)

We have used in [32] a very simple but rough technique: the information (or its opposite value, the entropy) of a random feature $\mathbf{f}$ is related to the log of the determinant of its covariance matrix. We can thus choose the most informative feature among the set to cluster and iteratively merge the closest feature to the current state estimate (according to the Mahalanobis distance). Each used feature is removed from the set. Once there are no more features to merge, we have obtained one cluster represented by its mean feature and we iterate the clustering stage on the remaining features. In this process, an efficient way of finding the nearest (Mahalanobis) neighbor would be an important improvement for the complexity. A more rigorous algorithm would be to let the different clusters compete with each other and to compute the mean feature of a cluster with a Mahalanobis distance minimization at each step.

5.3 Nearest Neighbor in a Manifold

This is an old problem and it is well studied on points in computational geometry. The corresponding notion is the *Voronoï diagram*. However, constructing and using this diagram is complex, so most techniques rely on space-partitioning methods such as k-D trees [36].

This last technique iteratively subdivides the space along each axis in turn and relies on the equivalence between the Euclidean L_2 norm and the L_∞ norm (maximum coordinate) which is separable along the axes. This allows us to give upper and lower bounds on the Euclidean norm with respect to the maximal difference between coordinates. Its generalization to a Riemannian manifold seems difficult since there no longer is a global coordinate system to define a L_∞ norm and alternate the search along the axes.

The Voronoï diagram can be generalized to a Riemannian manifold, but very little work exists. One can cite [4, chap. 18] for a hyperbolic manifold, but the determination of the diagram relies on the fact that the manifold has a negative curvature and thus there exists a global diffeomorphism with $\mathbb{R}^n$ (in this case a projection). These kinds of techniques are not applicable to positively curved manifolds such as spheres or projective spaces (which includes 3-D rotations). However, other techniques are possible. Watson developed a method to compute the Voronoï diagram on spheres $\mathcal{S}_n$ with their canonical Riemannian metric. Using the very special properties of the exponential charts, we think that it is possible to compute the Voronoï diagram on any homogeneous manifold with an invariant Riemannian metric. However, the efficiency of such a construction for the nearest neighbor problem is not ensured.

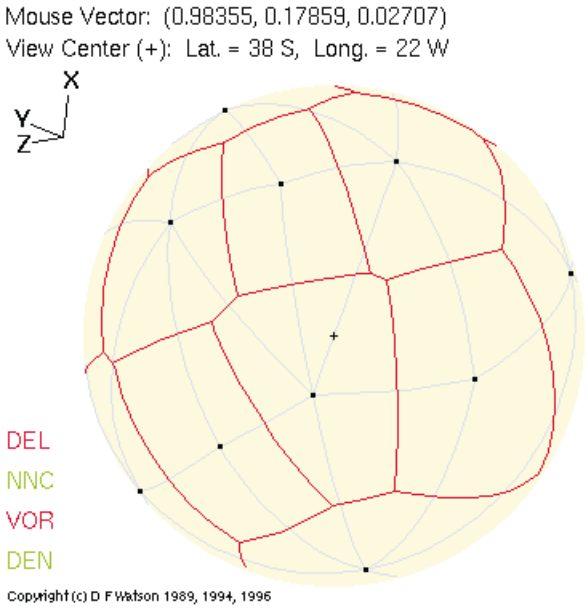


Figure 3. Voronoï diagram and dual Delaunay triangulation on the sphere $\mathcal{S}_2$, generated interactively and in real time with the java applet ModeMap of Dave Watson (*www.iinet.com.au/~watson/modemap.html*).

To conclude on this problem, we note that none of these techniques can apply for the nearest neighbor according to the Mahalanobis distance. Indeed, they rely on data preprocessing that makes use of the metric, which is only known at query time with the Mahalanobis distance. (It uses the covariance of the query feature.)

5.4 Uncertain Indexing

In all the algorithms that make use of invariants we have the same problem: how to retrieve efficiently the nearest or compatible invariants among a predefined set. The "brute force" method is to compare our query invariant with all the others, which has a linear complexity with respect to the number N of indexed invariants. There is no preprocessing stage, and the memory requirement is also $\mathcal{O}(N)$. There are basically three techniques to reduce the complexity. We have investigated in Section 5.3 the Voronoï diagram to find the nearest neighbor. We can subdivide the space sampling methods in two: either the sampling depends on the data, as in k-D trees, which generally gives a search time of $\mathcal{O}(\log N)$, or the space is sampled in a fixed way and the contents of a bin can be retrieved in quasi-constant time with a hash table. In this section we focus on this last technique.

In an ideal world with no noise on the data, we just have to determine a space sampling and a hash function that transform the coordinates of a bin into a code. This code indexes (through an array) the list of data having the same code. This hash function should scatter as much as possible the codes to obtain few empty lists and a very short mean list length. (Ideally there should be one and only one data per code.)

The introduction of the error on our invariants raises some problems in this algorithm: we now have to index or retrieve them using an error zone that generally intersects several bins in the sampled space. We have first to determine which bins. Next, all the bins create supplementary entries in the hash table. We will no longer have N entries but much more, for instance around $6N$ for the 2-D hash table of Figure 4. It is then important to quantify the mean number of indexed bins per data

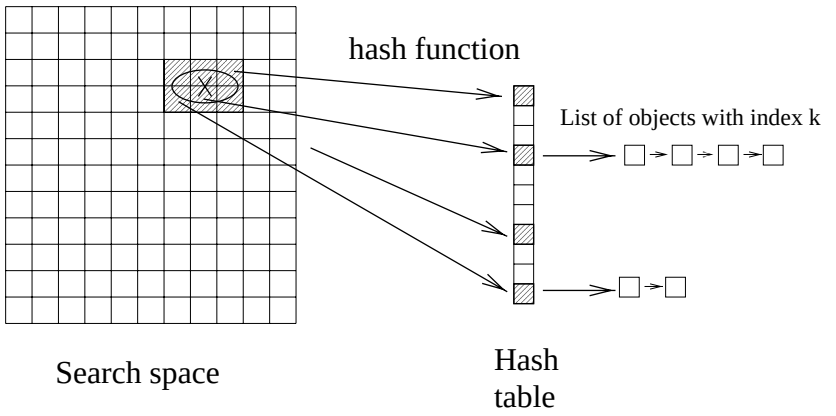


Figure 4. Multidimensional data hashing with error.

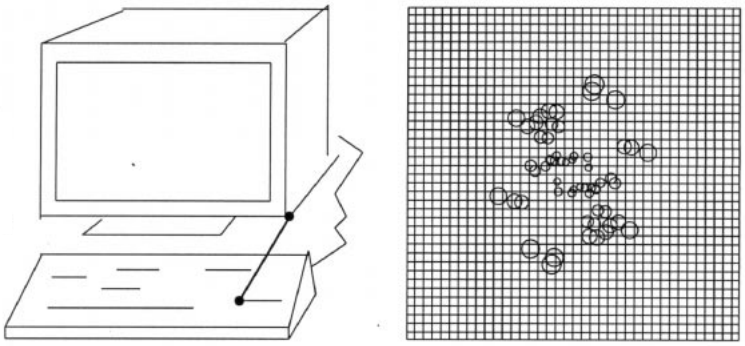


Figure 5. Left: a $512 \cdot 512$ (synthetic) 2-D image. The two points used for the basis are shown in bold with their error zone (five pixels). Right: the slice of the hash table corresponding to the measured length of the basis. One can see the sensitivity of the invariant coordinates by comparing the size of the error zones in the image space (left) and in the invariant coordinate space (right).

to have an idea of the mean number of objects by index (the mean length of the list of objects for a given code). The real complexity of hashing is proportional to these two factors. Since the hash function can associate the same code to very different locations and the volume of the compatibility zone is smaller than the indexed volume, it is moreover necessary to verify the compatibility of the query feature and those found in the corresponding bins of the hash table.

In [29], we have analyzed the complexity of the different types of hashing used by recognition algorithms on points. Let us index for instance points of $\mathbb{R}^d$ with an error zone bounded in norm by ε and a Cartesian sampling with width l (such as in Figure 5). The mean number of indexed bins by point (i.e., intersecting the error zone) is about $\bar{n} = (1 + 2 \cdot \varepsilon / l)^d$. Using this value in a false positives analysis, we concluded that $l \simeq \varepsilon$ was a good trade-off. Thus, for 3-D points, we have a mean number of nine indexed bins. For a fixed dimension, this is only a constant multiplicative factor in the complexity of the algorithm, but this factor is exponential with respect to the dimension. Hence, for the (rigid) geometric hashing of 3-D points, using the three invariants of the basis (the distance between the three points) and the coordinates of the fourth point in this basis, we index in a 6-D space and we obtain a mean number of index bins per point of 729!

Moreover, even if we assume the same metric error bound on all points, its propagation through the computation of invariants gives a bound that depends on the invariant position, as can be seen in Figure 5. Therefore, an adaptative space sampling is sometimes preferable.

Up to now, we have only raised the problems for the simple invariants of points. If we consider now a frame pair, the associated binary

invariant is also a frame. (It is in fact the rigid transformation from one frame to the other expressed in one of the frames.) The invariant space is this time $\mathcal{SO}_3 \times \mathbb{R}^3$ and no longer $\mathbb{R}^d$. The first question is how to sample regularly a space closing on itself such as the sphere or the set of rotations $\mathcal{SO}_3$. This problem is linked to the way we compute the intersection of the error zone with the bins, which has to be very efficient for the hashing to remain interesting.

The adaptation of hashing techniques to uncertain geometric features and invariants is thus a difficult problem, and, unless new techniques were to be designed, the speedup they were used for is not always conserved. Efficient searching for compatible features or invariants is, however, a crucial problem for the efficiency of recognition algorithms based on geometric features.

6 Performance Analysis: False Positives

With the correct handling of measurement errors, we have ensured the correctness of our recognition algorithms: there will be no (or very few) false negatives. On the other hand, we now have a larger probability of *false positives* (or phantom recognition). Indeed, using an error zone instead of a unique feature allows us to match features that fall by chance in this area, and, when this probability is sufficiently high, individual false matches can combine themselves (conspiracy) to produce an important matching score. We then will estimate that we have recognized an object that is not present.

There are two principal sources of false positives. The first one is that we are looking for a local consistency that can be insufficiently constrained. For instance, this is the case when we use only unary and binary invariants to predict global matches. A verification step is thus needed to insure the global consistency of the matches. This verification step can moreover improve the registration accuracy and rule out outliers. Since this step can be computationally expensive, we should limit the number of false positives as much as possible.

The second source of false positives is less obvious and comes from the simple modeling of the image (or the data) by features. Some important information can be dismissed in this process, and we can end up with a correct recognition from a feature point of view (a globally consistent matching) that is incorrect from the image or data point of view. This problem is inherent to the ascendent organization of information in image processing but can be minimized by using more-adapted and more-informative features. For instance, we will see in Section 6.2 that adding trihedra to 3-D points to form frames allows us to get rid of more than 80% of the individual false matches even with a very noisy trihedron (a standard deviation of 90°!). In real applications (registration of medical images from "extremal points" [34]), the standard deviation of the trihedron is about 10°, and the probability of false positives using frames drops off drastically with respect to points (Figure 7). In other experiments in molecular biology [32], we observed that using frames instead of points could rule out some biologically nonsensible matches.

In the following, we present a new method to analyze the probability of false positives for some matching methods. Then, we tackle a more general problem: the computation of the *intrinsic* complexity of the recognition problem, independently of the method used.

6.1 Problem Formulation

The probability of obtaining a false positive has been mainly studied in [15, 16, 27, 17] for recognition problems from a library of exact models based on points and lines. Here, we still consider that the model is exact but based on m geometric feature instead of points. The scene consists of n noisy features. In the model and in the scene, τ features are in the same configuration (up the noise). The $(m-\tau)$ and $(n-\tau)$ other features are supposed to be randomly distributed in the model image $\mathcal{I}_m$ and the scene image $\mathcal{I}_s$. In fact, if the meaning of this sentence is quite clear for points, it needs to be detailed for features. By "image," we mean the set of possible measurements of our features in the real image. For instance in a 3-D image, the position of a frame is constrained to be in the image volume $\mathcal{U} \subset \mathbb{R}^3$ but the trihedron part could be anything. Thus, the image (from a frame point of view) is $\mathcal{I} = \mathcal{SO}_3 \times \mathcal{U} \subset \mathcal{M}$. The random distribution of features in an "image" $\mathcal{I}$ is of course the uniform (i.e., invariant) distribution on this set.

To compute the false positives probability, we also need some hypotheses about the matching criterion. Here, we use one of the simplest: the criterion is the number of matches. A set of matches is accepted if its score after verification is greater than a given threshold. The matching algorithms being modified in the previous section not to miss a match, we can take τ for this threshold.

6.2 Selectivity: Probability of a (Single) False Match

Let f be a hypothesized transformation from the model to the scene: the exact feature x is matched with the random one $\mathbf{y}$ if the transformed feature $\mathrm{f} \star \mathrm{x}$ is in the error zone $\mathcal{Z}(\mathbf{y})$. This error zone can be based on a truncated probabilistic model (Equation 4) or on a bounded error model (Equation 5).

We call *selectivity* the probability of accepting this match if one of the features is an outlier (its position is random and uniform in the image). By symmetry, we assume that feature x is the outlier. Then, the probability of a false match is the conditional probability:

$$P(\mathrm{f} \star \mathrm{x} \leftrightarrow \mathbf{y}) = P\left((\mathrm{f} \star \mathrm{x}) \in \mathcal{Z}(\mathbf{y}) | \mathrm{x} \in \mathcal{I}_m\right).$$

The uniform distribution on the set $\mathcal{I}_m$ is given by the invariant measure $d\mathcal{M}$ on the feature manifold (see [33] for a discussion), normalized by the volume of the set $\mathcal{V}(\mathcal{I}_m) = \int_{\mathcal{I}_m} d\mathcal{M}$. Thus, the selectivity is

$$P(\mathrm{f} \star \mathrm{x} \leftrightarrow \mathbf{y}) = \int_{(\mathrm{f} \star \mathcal{I}_m) \cap \mathcal{Z}(\mathbf{y})} \frac{d\mathcal{M}}{\mathcal{V}(\mathcal{I}_m)} = \frac{\mathcal{V}\left((\mathrm{f} \star \mathcal{I}_m) \cap \mathcal{Z}(\mathbf{y})\right)}{\mathcal{V}(\mathcal{I}_m)}.$$

If the volume $\mathcal{V}(\mathcal{Z}(\mathbf{y}))$ of the error zone is small with respect to the image volume $\mathcal{V}(\mathcal{I}_m)$, we can consider that the transformed image $\mathrm{f}(\mathcal{I}_m)$ either contains the whole error zone or does not intersect it at all. This allows us to approximate the above probability by

$$P(\mathrm{f} \star \mathrm{x} \leftrightarrow \mathbf{y}) = \varepsilon \frac{\mathcal{V}\left(\mathcal{Z}(\mathbf{y})\right)}{\mathcal{V}(\mathcal{I}_m)} \quad \text{where} \quad \begin{cases} \varepsilon = 1 & \text{if } \mathrm{f}^{(-1)} \star \overline{\mathrm{y}} \in \mathcal{I}_m \\ \varepsilon = 0 & \text{otherwise.} \end{cases}$$

A desirable property for our "error volume" $\mathcal{Z}(\mathbf{y})$ is to be comparable at every point since we usually fix the same bound for error on all the points. This means that, for any feature $\mathbf{y}'$, there exists a transformation f such that $\mathbf{y}' = \mathrm{f} \star \mathbf{y}$ and $Z(\mathbf{y}') = \mathrm{f} \star \mathcal{Z}(\mathbf{y})$. The error volume is

said to be homogeneous. A stronger hypothesis is that for every transformation f, the error volume on the transformed point is the transformation of the error volume: $\mathcal{Z}(\mathrm{f} \star \mathbf{y}) = \mathrm{f} \star \mathcal{Z}(\mathbf{y})$. The volume is said to be isotropic in this case, and is completely determined by its shape around the origin. (See [35] for an analysis of noise models.) In both cases (homogeneity and isotropy), the volume of the error volume is invariant and can be computed at the origin. In the case of a homogeneous probabilistic model, it depends only on the covariance at the origin $\Sigma = J(\mathrm{f_y})^{(-1)} \cdot \Sigma_{\mathbf{yy}} \cdot J(\mathrm{f_y})^{-\mathrm{T}}$:

$$\mathcal{V}\left(\mathcal{Z}_\nu(\mathbf{y})\right) = \mathcal{V}_0 = \int_{\vec{\mathrm{x}}^{\mathrm{T}} \cdot \Sigma \cdot \vec{\mathrm{x}} \leq \nu^2} d\mathcal{M}(\vec{\mathrm{x}}).$$

Example with frames To keep this example as simple as possible, we consider here a bounded error model on the position and a separated one on the orientation. Thus, two frames are matched if the distance between their points is less than a threshold d_0 and if the rotation needed to adjust their trihedra has an angle less than a threshold θ_0 (this angle is $\theta = \|r_x^{(-1)} \circ r_y\|$). This error zone is isotropic. The volume is thus invariant and we can compute it at the origin: a frame $\mathrm{f} = (r, t)$ is in the error volume $Z(Id)$ if $\theta = \|r\| < \theta_0$ and $\|t\| < d_0$. Using the invariant measure on rigid transformations $d\mathcal{M}(r, t) = \frac{\sin^2(\|r\|/2)}{\|r\|^2}\, dr\, dt$, we can compute the volume of the error zone:

$$\mathcal{V}_0 = \int_{\theta<\theta_0} \int_{\|t\|<d_0} d\mathcal{M}(r, t) = \left(\int_{\theta<\theta_0} \frac{\sin^2(\theta/2)}{\theta^2}\, dr \right) \cdot \left(\int_{\|t\|<d_0} dt \right)$$

$$\mathcal{V}_0 = \left[2\pi(\theta_0 - \sin(\theta_0)\right] \cdot \left[\frac{4\pi}{3} d_0^3\right].$$

If we assume a cubic image of side l (256 for instance), this gives a Euclidean volume $V_I = l^3$ for points in which trihedra are not constrained: the rotation volume is $2\pi^2$. Finally, we obtain the basic probability of false match:

$$P(\mathrm{f} \star \mathrm{x} \leftrightarrow \mathbf{y}) = \varepsilon \cdot \eta \quad \text{with} \quad \eta = \left(\frac{\theta_0 - \sin\theta_0}{\pi}\right) \frac{4}{3} \left(\frac{d_0}{l}\right)^3.$$

We have isolated in the first term the probability of false match due to the trihedra only, which reflects the gain in selectivity when using frames instead of points. This function is plotted in Figure 6 and shows very interesting results: even for a bound of $\theta_0 = \pi/2 = 90$ deg, more than 80% of the random matches are rejected. For a more realistic bound of $\theta_0 = \pi/10 = 18$ deg, the probability of a false match drops to 0.0016: we would have to divide the bound on the position by 10 to obtain an equivalent selectivity using points only.

We will see in the rest of this section that it is sometimes useful to assume that the image is spherical, for instance with a diameter $d = \sqrt{3} \cdot l$. The volume of the (frame) image in this case is $\mathcal{V} = 4 \cdot \pi \cdot (d/2)^3/3$ and the selectivity becomes:

$$\eta = \left(\frac{\theta_0 - \sin\theta_0}{\pi}\right) \left(\frac{2 \cdot d_0}{d}\right)^3.$$

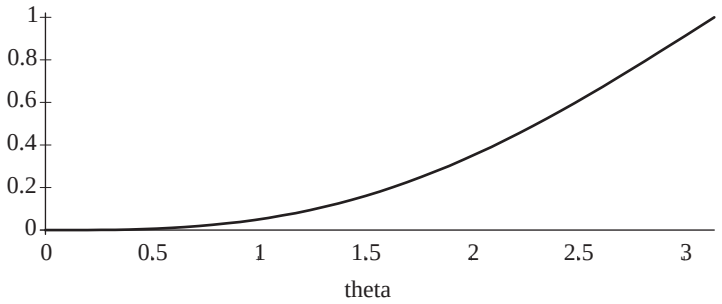


Figure 6. Basic probability of a false match for trihedra with a bound on the angle for the adjustment rotation of theta. Since the formula of the selectivity is multiplicative for frames, this curve is also the gain in selectivity when using frames instead of just points (the ratio of the selectivity of frames over the one of points).

6.3 Proportion of False Positives (Hough and Alignment)

In this section, we investigate some basic false positives analysis for the Hough transform and the alignment algorithm. Since many variants of these algorithms exist, the purpose is not to give a precise number of false positives but to explain with practical examples what techniques can be used. These techniques will be generalized in the next section to the analysis of the *intrinsic* number of false positives, independently of the algorithm used.

6.3.1 Probability of Choosing a "Correct Basis" in the Model

For the Hough transform and the alignment method, we begin by finding the possible matches of k features (k being the minimal number of matches to determine uniquely a transformation). By reference to the alignment algorithm, we call it a *hypothesis*. A hypothesis is correct if the k matches are all correct. One false match is sufficient to give an incorrect transformation.

There are $A_k^m = \binom{m}{k} \cdot k!$ possible sets of k ordinated features in the model, but only A_k^τ are correct. Thus, the probability of choosing an incorrect "basis" is (with an approximation for $k \ll \tau$):

$$p_{(m,k)} = 1 - \frac{A_k^\tau}{A_k^m} = 1 - \frac{\tau! \cdot (m-k)!}{m! \cdot (\tau - k!)} \simeq 1 - \left(\frac{\tau}{m}\right)^k .$$

6.3.2 Number of "Compatible Bases" in the Scene

Assume that we have chosen a basis in the model. A scene basis is compatible if there exists a transformation f that superimposes the k model basis features with the k scene basis features.

Let us investigate the case of a fixed (or known) transformation f. Assuming that the distribution of scene features is uniform, the probability that no features fall in one of the error zones of the transformed model basis is $(1 - \varepsilon \cdot \eta)^n$. Thus, the probability of finding at least one match in the scene for each model basis feature (under the transformation f) is

$$p_k(\mathrm{f}) = \prod_{i=1}^{k} (1 - (1 - \varepsilon_i \cdot \eta)^n) \quad \text{with} \quad \varepsilon_i = \begin{cases} 1 & \text{if } \mathrm{f} \star x_i \in \mathcal{I}_s \\ 0 & \text{otherwise.} \end{cases}$$

Now, we just have to integrate over all possible transformations to obtain the mean number of compatible bases in the scene. We note that

the expression above is null if one of the ε_i is null. Thus, we can rewrite it as $p_k(\mathrm{f}) = (1-(1-\eta)^n)\prod_{i=1}^{k}\varepsilon_i$ and the integral becomes:

$$P_k = \int_{\mathrm{f}\in\mathcal{G}} p_k(\mathrm{f})\cdot d\mathcal{G} = (1-(1-\eta)^n)^k \cdot \int_{\mathrm{f}\in\mathcal{G}} \prod_{i=1}^{k} \varepsilon_i(\mathrm{f})\cdot d\mathcal{G}.$$

An upper bound of the value of the right term α is obtained by only asking for the intersection of the images:

$$\prod_{i=1}^{k} \varepsilon_i(\mathrm{f}) = 1 \qquad \Longrightarrow \qquad \mathrm{f}\star\mathcal{I}_m \cap \mathcal{I}_s \neq \emptyset.$$

In the case of 3-D rigid transformations, we can obtain a rough estimation of this bound as follows: let d be the image diameter (for instance $d=\sqrt{3}\cdot l$ for a cubic image of side l). Assuming the images are spherical, and taking the origin at the center, we can make any rotation followed by any translation of length less than d and the images still intersect. On the other hand, any translation of length greater than d separates the two images. Thus, we can bound our integral by

$$\alpha \leq 2\cdot\pi^2\cdot\int_{\|t\|<d} dt = \frac{8}{3}\cdot\pi^3\cdot d^3 = \frac{(2\cdot\pi\cdot d)^3}{3} = (2\cdot\pi\cdot l)^3.$$

Thus, given a model basis, the mean number (nonnormalized probability) of corresponding bases in the scene image is

$$P_k = \alpha\cdot(1-(1-\eta)^n)^k = \alpha\cdot(n\cdot\eta)^k \quad \text{with } \alpha \leq \frac{(2\cdot\pi\cdot d)^3}{3}.$$

6.3.3 Hough Transform: Proportion of False Transformations

Ideally, we should compute the mean number of true and false hypotheses in each bin of the transformation space. To simplify the problem, we just compute the percentage of good hypotheses. Roughly, this amounts to assuming a uniform distribution of both good and false hypotheses in the transformation space.

We have A_k^τ correct hypotheses. But for each (correct or incorrect) model base, we find a mean number of P_k false hypotheses, which give $P_k\cdot A_k^m$ false hypotheses. Thus, the proportion of false transformations in the accumulator is

$$\begin{aligned} P_{Hough} &= \frac{P_k\cdot A_k^m}{A_k^\tau + P_k\cdot A_k^m} \\ &= \left(1+\frac{A_k^\tau}{P_k\cdot A_k^m}\right)^{(-1)} \simeq \left(1+\frac{1}{\alpha}\left(\frac{\tau}{\eta\cdot m\cdot n}\right)^k\right)^{(-1)}. \end{aligned}$$

6.3.4 Probability of Accepting a Verification

For the alignment algorithm, we have to verify each hypothesis. Given a hypothesis, k features are already used to constitute the scene basis. Thus, there are $n-k$ error zones in the scene. The probability that one of the $m-k$ model features avoids all the error zones is $(1-\eta)^{n-k}$. Thus, the probability that it falls in at least one the error zone is

$$p = 1-(1-\eta)^{n-k} \simeq (n-k)\cdot\eta.$$

The probability to match exactly j features among the $m-k$ model features is the binomial

$$B_{(m-k,p)}(j) = \binom{m-k}{j} p^j (1-p)^{m-k-j}.$$

Thus, the probability of accepting a hypothesis is the sum of the probabilities to have more than τ matches:

$$\begin{aligned} q &= \sum_{j \geq \tau} B_{(m-k,p)}(j) = 1 - \sum_{j=0}^{\tau} \binom{m-k}{j} p^j (1-p)^{m-k-j} \\ &= I_p(\tau+1, m-\tau), \end{aligned}$$

where $I_p(a,b)$ is the normalized incomplete Beta function.

Assuming that $p \ll 1$, we can approximate the binomial $B_{(m-k,p)}$ by the Poisson distribution of parameter $\lambda = (m-k) \cdot p$:

$$q \simeq 1 - e^{-\lambda} \sum_{j=0}^{\tau} \frac{\lambda^j}{j!} \quad \text{with} \quad \lambda \simeq (n-k) \cdot (m-k) \cdot \eta.$$

6.3.5 Alignment: Proportion of False Positives

We have A_k^m possible bases in the model, and a mean number of P_k false hypotheses for each one, each hypothesis being accepted with a probability q. Thus, we accept a mean number of $A_k^m \cdot P_k \cdot q$ incorrect matchings. We also have A_k^τ correct hypotheses that will give correct matchings. Thus, the proportion of false positives is

$$\begin{aligned} P_{align} &= \frac{q \cdot P_k \cdot A_k^m}{A_k^\tau + q \cdot P_k \cdot A_k^m} \\ &= \left(1 + \frac{A_k^\tau}{q \cdot P_k \cdot A_k^m}\right)^{(-1)} \simeq \left(1 + \frac{1}{\alpha \cdot q}\left(\frac{\tau}{\eta \cdot m \cdot n}\right)\right)^{(-1)}. \end{aligned}$$

The probability q can be directly interpreted as the filtering ratio of the verification step.

6.4 Intrinsic *Probability of False Positives*

In the previous section, we try to characterize the way two algorithms act by propagating step by step the probability of incorrect matches. One can also ask for the *intrinsic* probability of false positives, independently of the algorithm used: what is the probability to obtain a score τ if the model and the scene are in fact constituted of m and n features randomly (uniformly) distributed in the images?

In fact, we have developed in the previous section the basic tools to answer this question. Indeed, finding τ matches means that there exists a transformation f such that τ model features fall in the error zones of the scene features. Thus, we just have to compute the probability of obtaining τ matches for a given transformation, and then integrate this probability for all possible transformations.

We have n error zones in the scene. Given a transformation f, the probability for one model feature to fall in at least one of these zones is $p \simeq \eta \cdot n$. Thus, the probability that exactly j of the m model features fall

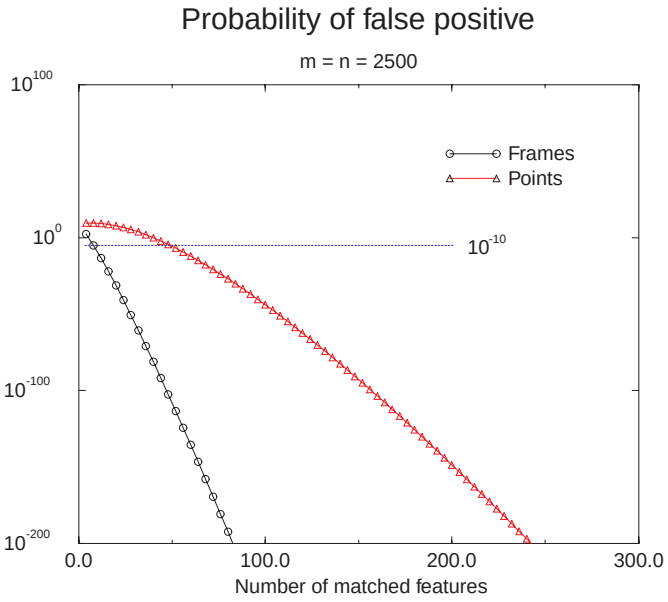


Figure 7. Qualitative estimation of the number of false positives involving at least x matches in MR images of 2,500 features. Comparison between frames and points: we need roughly five times more point matches than frame matches to obtain the same probability (10 frames and 56 point matches for a probability of 10^{-10}). The actual match found involves about 500 features, and its probability of being a false positive is thus practically zero.

in a scene error zone is the binomial $B_{(m,p)}(j)$. Therefore, the probability of matching at least τ of them is

$$q = \sum_{j \geq \tau} B_{(m,p)}(j) = I_p(\tau + 1, m) \simeq 1 - e^{-\lambda} \sum_{j=0}^{\tau} \frac{\lambda^j}{j!},$$

$$\text{with} \quad \lambda = m \cdot p = n \cdot m \cdot \eta.$$

To obtain the mean number of false positives, we just have to integrate over all possible transformations, which amounts to multiplying it by α. Finally, with our spherical image of diameter d, the probability of matching τ features among m and n random features is

$$\Phi = \alpha \cdot q \simeq \frac{(2 \cdot \pi \cdot d)^3}{3} \cdot \left(1 - e^{-\lambda} \sum_{j=0}^{\tau} \frac{\lambda^j}{j!} \right) \quad \text{with} \quad \lambda = n \cdot m \cdot \eta. \quad (6)$$

This equation is valid only when $\lambda = n \cdot m \cdot \eta \ll 1$. Moreover, as our analysis is very qualitative, we can trust it only for very small values. However, notice that the influence of the integration over admissible transformations on the number of false positives is linear. This is to be compared with the exponential influence of the selectivity η in the accepting probability q. Hence, the estimation of α does not need to be really accurate (or could be adjusted a posteriori from experiments). This effect is visible in Figure 7: the difference in selectivity between points and frames induces very different false positives probabilities, whereas a variation on α would produce only a small vertical shift of the curves in logarithmic scale.

6.4.1 Application Example in Medical Imaging

In the case of the MR images of the associated demo, the volume images are $256 \cdot 256 \cdot 165$ mm and we typically extract 2,500 extremal points (modeled by semi-oriented frames). Among them, about 500 are matched in a registration.

Frame selectivity Combining the error on the model and on the scene features, we obtain a roughly diagonal covariance matrix:

$$\Sigma = \text{DIAG}(0.005, 0.006, 0.065; 0.45, 0.60, 0.165).$$

On such a small error zone, we can approximate the invariant measure on frames by the Lebesgue measure:

$$\frac{\sin^2(\theta/2)}{\theta^2} \cdot d\theta \cdot dt \simeq (1 + O(\theta^2)) \cdot \frac{d\theta \cdot dt}{4}.$$

Thus, the volume of the error zone defined by a χ^2 limit of ν^2 is

$$\mathcal{V}_0 = \int_{x^{\mathrm{T}} \cdot \Sigma^{(-1)} \cdot x \leq \nu^2} \frac{dx}{4}.$$

To compute this integral, we make the change of variable $x = \nu \cdot \Lambda \cdot y$, where Λ is a square root of Σ. We get

$$\mathcal{V}_0 = \frac{1}{4} \cdot \int_{y^{\mathrm{T}} \cdot y < 1} \nu^6 \cdot |\det(\Lambda)| \cdot dy = \nu^6 \cdot \sqrt{\det(\Sigma)} \cdot \frac{\pi^3}{6} \cdot \frac{1}{4}.$$

For a χ^2 bound of $\nu^2 = 16$, this gives $\mathcal{V}_0 \simeq 1.56$.

The total volume of the image is $256 \cdot 256 \cdot 165 = 1.0810^7$ for the frame position and $\int_{\|r\| \leq \pi} \frac{\sin^2(\|r\|/2)}{\|r\|^2} \cdot dr = 2 \cdot \pi^2$ for the frame trihedron. In fact, extremal points are modeled using semi-oriented frames, and we have to divide this volume by 2. Thus the image volume is

$$\mathcal{V}(\mathcal{I}) = 256 \cdot 256 \cdot 165 \cdot \pi^2 = 1.067\ 10^8$$

Thus, the selectivity is: $\eta = \frac{\mathcal{V}_0}{\mathcal{V}(\mathcal{I})} = 1.46 \times 10^{-8}$.

Point selectivity In the case of points, the noise model becomes isotropic with a standard deviation $\sigma = 0.50$. We use a 3-D χ^2 of $\nu^2 = 6$. The volume of the error zone is now $\mathcal{V}_0 = \nu^3 \cdot \sigma^3 \cdot \frac{4}{3} \cdot \pi \simeq 21.7$, and the image volume is $\mathcal{V}(\mathcal{I}) = 1.08 \times 10^7$. Thus, we obtain a selectivity of $\eta_{pt} = 2.01 \times 10^{-6}$. This is a loss of two orders of magnitude with respect to frames!

Probability of false positives To obtain an upper bound on the number of false positives, we take as image diameter $d = \sqrt{3} \cdot l$ with $l = 256$, which gives an integration factor $\alpha = (2 \cdot \pi \cdot l)^3 = 4.16 \times 10^9$. The number of features is $n = m = 2{,}500$. We plot in Figure 7 the probability of a random match of τ features using Equation 6. In both cases (frames and points), the probability of a false positive involving 500 matches (the observed value in real images) is null at the machine accuracy. However, the difference between points and frames is visible for smaller sets of matches. For instance, if we decide that a match is not a false positive if its probability to be so is less than $\Phi = 10^{-10}$, we accept matches made of at least 10 frames or 56 points. In fact, we need roughly five times more matches using points than using frames to obtain the same probability of false positives! The interest of using frames instead of points is clear.

Moreover, complex geometric features do have more invariants than simple features, which can considerably reduce the complexity of matching. We were for instance able to reduce the complexity of the 3-D substructure matching problem from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^2)$ using frames instead of points. Using geometric features more informative than points is thus crucial to increase the robustness of recognition algorithms and reduce their complexity.

6.4.2 Application Example in Molecular Biology

In [32], we use frames to model amino acids in proteins. (See the html demo.) In the example of this article, the radius of the protein is about 20 Å, which gives an "image volume" of $\mathcal{V}_{\mathcal{I}} = \frac{4}{3} \cdot \pi \cdot r^3 \cdot (2 \cdot \pi^2) \simeq 661500$ for frames and $\mathcal{V}_{\mathcal{I}} = \frac{4}{3} \cdot \pi \cdot r^3 \simeq 33510$ if we use just points. We use a diagonal covariance matrix with standard deviations $\sigma_r = 15^o = 0.15$ rad for the trihedron and $\sigma_t = 0.35$ Å for the position. The χ^2 values are 16 for frames and 8 for points. Thus, the volume of the error zones are $\mathcal{V}_0 = \chi^6 \cdot \sigma_r^6 \cdot \sigma_t^6 \cdot \frac{\pi^3}{24} = 4.8 \times 10^{-2}$ for frames and $\mathcal{V}_0 = \chi^3 \cdot \sigma_t^3 \cdot \frac{4 \cdot \pi}{3} = 4.06$. Finally, we obtain the following selectivities:

$$\eta_{fr} = 7.26 \times 10^{-7} \quad \text{and} \quad \eta_{pt} = 1.2 \times 10^{-4}.$$

The integration factor alpha is $\alpha = (8 \cdot \pi \cdot r)^3/3 = 4.2 \times 10^7$, and we have 65 and 105 amino acids in the two proteins. With these values, we need to match at least 5 frames or 20 points to obtain a probability of false match less than 10^{-10}, and the main substructures we found in common score 22, 16, and 13 matches. In this case, using frames is critical to ensure the significance of our matching results.

6.5 Discussion: Using the False Positives Analysis to Improve the Algorithms

When we tackle a new problem, the first application of this analysis is to provide the intrinsic complexity of the recognition problem. Depending on the type, the error model, and the number of features used, we can compute the threshold τ on the number of matches to use in order to have a given probability of false positives (for instance $\Phi = 10^{-10}$). If this threshold is well above the number of matches we expect, we can proceed and choose an algorithm with a low-complexity basis. Otherwise (as in the above example with proteins), we expect any algorithm to reach its maximal complexity and possibly give false positives. In this case, a solution is to model the images by more-complex geometric features, i.e., to take into account more information in the feature-based recognition process.

However, we should notice that the false positive analysis we presented is based on several limitative assumptions (exact model, uniform distribution of features, etc.) which are hardly ever verified in real cases. For instance, in medical images of the head, extremal points are not uniformly distributed in the image, but more or less uniformly distributed on the surface of the brain and the skull, which can be roughly considered as spheres. Thus, the performance analysis we presented should not be considered as quantitative estimations but only as indicative of a general behavior.

A very interesting extension would be to compute the probability of false positives *online*, during the recognition algorithm itself. This would allow us to take into account the specific distribution of the model and scene features. In our generic framework for feature-based recognition, this could be done by adding a basic operation on features computing the selectivity of matches using the covariance matrix and other operations to propagate this probability through the computations, as we did for the uncertainty. Doing so, we could evaluate the quality of a match at any stage of the algorithm and give up or delay the processing of bad hypotheses. This should be related with probabilistic matching techniques developed in [38].

As far as a specific recognition algorithm is concerned (for instance the Hough transform or the alignment method), we could extend the false positive analysis to compute the mean number of hypotheses to verify before finding a good one, the mean complexity of the algorithms, etc. However, these computations rely on the specific settings of these algorithms, and we cannot develop the analysis for all approaches: we leave it to the reader to analyze his own algorithm. We believe that such an analysis may help to discriminate between different options or settings of an algorithm depending on the application context.

7 Conclusion

We have formalized in this article the main matching algorithms in terms of geometric features and showed how to modify them in order to incorporate explicitly and rigorously the uncertainty of measurements. The drawback of the correctness of the algorithms is the presence of false positives. We developed a new method to analyze the probability of false positives in two particular algorithms and generalized it to the evaluation of the *intrinsic* complexity of the matching problem, independently of the method used. Doing so, we showed that using more-informative features, such as frames instead of points, can drastically reduce this probability and allow for the use of lower-complexity matching methods.

For a computational point of view, we have identified four still-open problems to implementing these generic matching algorithms. The first one is linked to the structure of the space of n-ary invariants, and we have formalized it with the shape-space theory. We believe that it is not only possible to characterize, in that way, the manifold of invariants but also a suitable metric structure. The second problem concerns the clustering of features on a manifold, and more particularly with uncertainty information. The method we use to solve this problem can certainly be improved. The last two key points are linked to the efficiency of searching algorithms. This is for the first part the search of the nearest neighbor, with respect to the Riemannian metric or the Mahalanobis distance, and for the other part the search for all the compatible features or invariants in the χ^2 sense. We have observed that hashing techniques raise some important problems because of measurement uncertainty and of the non-Euclidean topology of the concerned manifolds. These problems turn crucial with invariants of features with high dimension. There is perhaps a trade-off to be found between adaptative space sampling and hashing methods.

In conclusion, generic, correct, and robust recognition algorithms on uncertain geometric features are possible. However, from a computational point of view, we need to design speed-up techniques for efficiency.

References

[1] Ayache N. and Faugeras, O. D. Hyper : A new approach for the recognition and positioning of two-dimensional objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(1):44–54, 1986.

[2] Besl, P. J. and Jain, R. C. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, March 1985.

[3] Besl, P. J. and McKay, N. A method for registration of 3-D shapes. *PAMI*, 14(2):239–256, 1992.

[4] Boissonnat, J. D. and Yvinec, M. *Géométrie algorithmique*. Ediscience international, Paris, 1995.

[5] Breuel, T. M. *Geometric Aspects of Visual Object Recognition*. Doctoral dissertation, AI-TR 1374, MIT, 1992.

[6] Califano, A. and Mohan, R. Multidimensional indexing for recognizing visual shapes. In *Proc. CVPR 91*, pages 28–34, June 1991.

[7] do Carmo, M. *Riemannian Geometry*. Mathematics. Birkhäuser, Boston, Basel, Berlin, 1992.

[8] Chin, R. T. and Dyer, C. R. Model-based recognition in robot vision. *ACM Computing Surveys*, 18(1):67–108, March 1986.

[9] Duda, R. O. and Heart, P. E. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[10] Faugeras, O. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.

[11] Fischer, D., Bachar, O., Nussinov, R., and Wolfson, H. An efficient automated computer vision-based technique for detection of three-dimensional structural motifs in proteins. *J. of Biomolecular Structures and Dynamics*, 9(4):769–789, 1992.

[12] Fischer, O., Nussinov, R., and Wolfson, H. 3-D substructure matching in protein molecules. In *Combinatorial Pattern Matching 92—LNCS 644*, pages 136–150. Springer Verlag, 1992.

[13] Grimson, W. E. L. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, 1990.

[14] Grimson, W. E. L. and Huttenlocher, D. P. On the sensitivity of geometric hashing. In *Proc. Third ICCV*, pages 334–338, 1990.

[15] Grimson, W. E. L. and Huttenlocher, D. P. On the sensitivity of the Hough transform for object recognition. *IEEE PAMI*, 12(3):255–274, March 1990.

[16] Grimson, W. E. L. and Huttenlocher, D. P. On the verification of hypothesized matches in model-based recognition. *IEEE PAMI*, 13(12):1201–1213, December 1991.

[17] Grimson, W. E. L., Huttenlocher, D. P., and Jacobs, D. W. A study of affine matching with bounded sensor error. *Int. Journ. of Comput. Vision*, 13(1):7–32, 1994.

[18] Guéziec, A. and Ayache, N. Smoothing and matching of 3d space curves. *Intern. Journ. of Comput. Vision*, 12(1):79–104, 1994.

[19] Guéziec, A., Pennec, X., and Ayache, N. Medical image registration using geometric hashing. *IEEE Computational Science & Engineering, special issue on Geometric Hashing*, 4(4):29–41, October–December 1997.

[20] Huttenlocher, D. P. and Ullman, S. Object recognition using alignment. In *Proc. of ICCV*, pages 72–78, 1987.

[21] Huttenlocher, D. P. and Ullman, S. Recognizing solid objects by alignment with an image. *Int. Journ. Computer Vision*, 5(2):195–212, 1990.

[22] Jain, A. K. and Dubes, R. C. *Algorithms for clustering data*. Englewood Cliffs, N.J., 1988.

[23] Kendall, D. G. A survey of the statistical theory of shape (with discussion). *Statist. Sci.*, 4:87–120, 1989.

[24] Kishon, E., Hastie, T., and Wolfson, H. J. 3-D curve matching using splines. *Journal of Robotic Systems*, 8(6):723–743, 1991.

[25] Klingenberg, W. *Riemannian Geometry*. Walter de Gruyter, Berlin, New York, 1982.

[26] Lamdan, Y. and Wolfson, H. J. Geometric hashing : A general and efficient model-based recognition scheme. In *Proc. of Second ICCV*, pages 238–289, 1988.

[27] Lamdan, Y and Wolfson, H. J. On the error analysis of geometric hashing. In *IEEE Int. Conf. on Comput. Vis. and Patt. Recog.*, pages 22–27, 1991.

[28] Le, H. and Kendall, D. G. The Riemannian structure of Euclidean shape space: A novel environment for statistics. *Ann. Statist.*, 21:1225–1271, 1993.

[29] Pennec, X. Correctness and robustness of 3-D rigid matching with bounded sensor error. Research Report 2111, INRIA, November 1993.

[30] Pennec, X. Registration of uncertain geometric features: Estimating the pose and its accuracy. In *Proc of the First Image Registration Workshop,* November 20–21, 1997, Greenbelt, Maryland, USA. CESDIS, 1997.

[31] Pennec, X. Computing the mean of geometric features: Application to the mean rotation. Research Report 3371, INRIA, March 1998.

[32] Pennec, X. and Ayache, N. A geometric algorithm to find small but highly similar 3D substructures in proteins. *Bioinformatics*, 14(5), 1998. In press.

[33] Pennec, X. and Ayache, N. Uniform distribution, distance and expectation problems for geometric features processing. *Journal of Mathematical Imaging and Vision*, 9(1), 1998.

[34] Pennec, X. and Thirion, J. P. A framework for uncertainty and validation of 3D registration methods based on points and frames. *Int. Journal of Computer Vision*, 25(3):203–229, 1997.

[35] Pennec, X. *L'Incertitude dans les Problèmes de Reconnaissance et de Recalage: Applications en Imagerie Médicale et Biologie Moléculaire*. Doctoral dissertation, Ecole Polytechnique, Palaiseau (France), December 1996.

[36] Preparata, F. P. and Shamos, M. I. *Computational Geometry, an Introduction*. Springer Verlag, 1985.

[37] Rigoutsos, I. *Massively Parallel Baysian Object Recognition*. Doctoral dissertation, New York University, August 1992.

[38] Rigoutsos, I. and Hummel, R. A Bayesian approach to model matching with geometric hashing. *Computer vision and image understanding: CVIU*, 62(1):11–26, July 1995.

[39] Spivak, M. *Differential Geometry*, volume 1. Publish or Perish, Inc., 2nd edition, 1979.

[40] Stein, F. and Medioni, G. G. Structural hashing: Efficient three-dimensional object recognition. *PAMI*, 14(2):125–145, February 1992.

[41] Wolfson, H. J. Model-based recognition by geometric hashing. In O. Faugeras, editor, *Proc. of 1st Europ. Conf. on Comput. Vision (ECCV 90)*, Lecture Notes in Computer Science 427, pages 526–536, Springer Verlag, April 1990.

[42] Zhang, Z. Iterative point matching for registration of free-form curves and surfaces. *Int. Journ. Comp. Vis.*, 13(2):119–152, 1994.